Faster sublinear approximation of the number of k-cliques in low-arboricity graphs

Talya Eden * Dana Ron † C. Seshadhri ‡

Abstract

Given query access to an undirected graph G, we consider the problem of computing a $(1 \pm \varepsilon)$-approximation of the number of k-cliques in G. The standard query model for general graphs allows for degree queries, neighbor queries, and pair queries. Let n be the number of vertices, m be the number of edges, and n_k be the number of k-cliques. Previous work by Eden, Ron and Seshadhri (STOC 2018) gives an $O^*(\frac{n}{n_k^{1/k}} + \frac{m^{k/2}}{n_k})$-time algorithm for this problem (we use $O^*(\cdot)$ to suppress $\mathrm{poly}(\log n, 1/\varepsilon, k^k)$ dependencies). Moreover, this bound is nearly optimal when the expression is sublinear in the size of the graph.

Our motivation is to circumvent this lower bound, by parameterizing the complexity in terms of *graph arboricity*. The arboricity of G is a measure for the graph density "everywhere". There is a very rich family of graphs with bounded arboricity, including all minor-closed graph classes (such as planar graphs and graphs with bounded treewidth), bounded degree graphs, preferential attachment graphs and more.

We design an algorithm for the class of graphs with arboricity at most α, whose running time is $O^*(\min\{\frac{n\alpha^{k-1}}{n_k}, \frac{n}{n_k^{1/k}} + \frac{m\alpha^{k-2}}{n_k}\})$. We also prove a nearly matching lower bound. For all graphs, the arboricity is $O(\sqrt{m})$, so this bound subsumes all previous results on sublinear clique approximation.

As a special case of interest, consider minor-closed families of graphs, which have constant arboricity. Our result implies that for any minor-closed family of graphs, there is a $(1 \pm \varepsilon)$-approximation algorithm for n_k that has running time $O^*(\frac{n}{n_k})$. Such a bound was not known even for the special (classic) case of triangle counting in planar graphs.

1 Introduction

The problem of counting the number of k-cliques in a graph is a fundamental problem in theoretical computer science [10, 49, 40, 58, 7], with a wide variety of applications [37, 13, 53, 18, 44, 8, 6, 31, 54, 38, 27, 57, 30, 39]. This problem has seen a resurgence of interest because of its importance in analyzing massive real-world graphs (like social networks and biological networks). There are a number of clever algorithms for exactly counting k-cliques using matrix multiplications [49, 26] or combinatorial methods [58]. However, the complexity of these algorithms grows with $m^{\Theta(k)}$, where m is the number of edges in the graph.

A line of recent work has considered this question from a sublinear approximation perspective [20, 24]. Letting n denote the number of vertices, m the number of edges, and n_k the number of k-cliques, the complexity of approximating the number of k-cliques up to a $(1\pm\varepsilon)$-multiplicative factor is $O^*\left(\frac{n}{n_k^{1/k}} + \frac{m^{k/2}}{n_k}\right)$ with a nearly matching lower bound [24].[1]

We study the problem of approximating the number of k-cliques in bounded arboricity graphs, with the hope of circumventing the above lower bound.[2] A graph of arboricity at most α has the property that the average degree in any subgraph is at most 2α [46, 47]. One of our motivations is to understand when it is possible to get a running time of $O^*(n/n_k)$. This is an obvious lower bound, since a graph can simply contain n_k disjoint k-cliques, and, e.g., a cycle on the remaining vertices. It requires $\Omega(n/n_k)$ uniform vertex samples just to land in a k-clique. Are there classes of graphs for which one can accurately estimate the number of k-cliques in this time?

A consequence of our main theorem is an affirmative answer to this question, for the class of constant-arboricity graphs. The class of graphs with constant arboricity is an immensely rich class, containing, among others, all minor-closed graph families. The concept of constant arboricity plays a significant role in the theory of *bounded expansion* graphs, which has applications

*CSAIL at MIT, *talyaa01@gmail.com*. The majority of this work was done while the author was affiliated with Tel Aviv University. This research was partially supported by a grant from the Blavatnik fund, and Schmidt and Rothschild Fellowships. The author is grateful to the Azrieli Foundation for the award of an Azrieli Fellowship.

†Tel Aviv University, *danaron@tau.ac.il*. This research was partially supported by the Israel Science Foundation grants No. 671/13 and 1146/18.

‡University of California, Santa Cruz, *sesh@ucsc.edu*. This research was funded by NSF CCF-1740850, NSF CCF-1813165, and ARO Award W911NF191029.

[1] As stated in the abstract, we use the $O^*(\cdot)$ notation to suppress $\mathrm{poly}(\log n, 1/\varepsilon, k^k)$ dependencies.

[2] The arboricity of a graph is the minimal number of forests required to cover the edges of the graph.

in logic, descriptive complexity, and fixed parameter tractability [48]. In the context of real-world graphs, the classic Barabási-Albert preferential attachment graphs as well as additional models generate constant arboricity graphs [3, 5, 4]. In most real-world graphs, the arboricity is at most an order of magnitude larger than the average degree, while the maximum degree is three to four orders of magnitude larger [32, 39, 55]. In practical applications, low arboricity is often exploited for faster algorithms for clique and dense subgraph counting [28, 30, 45, 39, 16].

A classic result of Chiba and Nishizeki gives an $O(n+m\alpha^{k-2})$ algorithm for exact counting of k-cliques in graphs of arboricity at most α [10]. Our primary motivation is to get a sublinear-time algorithm for approximating the number of k-cliques on such graphs. We assume the standard query model for general graphs (refer to Chapter 10 of Goldreich's book [33]), so that the algorithm can perform degree, neighbor and pair queries. Let us exactly specify each query. (1) Degree queries: given $v \in V$, get the degree $d(v)$. (2) Neighbor queries: given $v \in V$ and $i \leq d(v)$ get the i^{th} neighbor of v. (3) Pair queries: given vertices u, v, determine if (u, v) is an edge.[3]

1.1 Results. Our main result is an algorithm for approximating the number of k-cliques, whose complexity depends on the arboricity. The algorithm is sublinear for $n_k = \omega(\alpha^{k-2})$ (and we subsequently show that for smaller n_k, sublinear complexity cannot be obtained).

THEOREM 1.1. *There exists an algorithm that, given n, k, an approximation parameter $0 < \varepsilon < 1$, query access to a graph G, and an upper bound α on the arboricity of G, outputs an estimate $\widehat{n}_k$, such that with high constant probability (over the randomness of the algorithm),*

$$(1-\varepsilon)\cdot n_k \leq \widehat{n}_k \leq (1+\varepsilon)\cdot n_k.$$

The expected running time of the algorithm is

$$\min\left\{\frac{n\alpha^{k-1}}{n_k},\ \frac{n}{n_k^{1/k}}+\frac{m\alpha^{k-2}}{n_k}\right\}\cdot \mathit{poly}(\log n, 1/\varepsilon, k^k),$$

and the expected query complexity is the minimum between the expected running time and $O(m+n)$.

[3] Gonen et al. [35] proved that any algorithm for approximating the number of triangles when given access only to degree and neighbor queries, requires $\Omega(n)$ queries when $m = \Theta(n)$. We note that their lower bound is based on constructing two families of graphs, where both families have constant arboricity. Therefore, their lower bound holds for bounded arboricity graphs.

Recall that α is always upper bounded by $\sqrt{m}$, so that the bound in Theorem 1.1 subsumes the result for for approximating the number of k-cliques in general graphs [24]. As we discuss in more detail in Section 2, our algorithm starts similarly to the algorithm of [24] but departs quickly since it relies on a different, iterative, approach so as to achieve the dependence on α.

Comparing our bound of $O^*\left(\frac{n}{n_k^{1/k}}+\frac{m\alpha^{k-2}}{n_k}\right)$ for approximate counting with the Chiba and Nishizeki bound of $O(n+m\alpha^{k-2})$ for exact counting, we get that when $n_k \gg \text{poly}\left(\frac{\log n\cdot k^k}{\varepsilon}\right)$, our bound is smaller, and as n_k increases the gap becomes more significant. Note that Chiba and Nishizeki read the entire graph, so that they have full knowledge of the graph, and their challenge is to count the number of k-cliques (by enumerating them), as efficiently (in terms of running time) as possible. On the other hand, our algorithm may obtain only a partial view of the graph. Hence, our challenge is to compute an estimate of the number of k-cliques based on such partial knowledge, by devising a careful sampling procedure (that in particular, exploits the bounded arboricity).

An application of Theorem 1.1 for the family $\mathcal{G}$ of minor-closed graphs[4] gives the following corollary. We note that even for the special case of *triangle* counting in planar graphs, such a result was not previously known.

COROLLARY 1.2. *Let $\mathcal{G}$ be a minor-closed family of graphs. There is an algorithm that, given n, k, ε, and query access to $G \in \mathcal{G}$, outputs a $(1\pm\varepsilon)$-approximation of n_k with high constant probability. The expected running time of the algorithm is*

$$(n/n_k)\cdot \mathit{poly}(\log n, 1/\varepsilon, k^k).$$

In general, we prove that the bound of Theorem 1.1 is nearly optimal.

THEOREM 1.3. *Consider the set $\mathcal{G}$ of graphs of arboricity at most α. Any multiplicative approximation algorithm that succeeds with constant probability on all graphs in $\mathcal{G}$ must make*

$$\Omega\left(\min\left\{\frac{n\alpha^{k-1}}{k^k\cdot n_k},\ \frac{n}{k\cdot n_k^{1/k}}\right\}+\min\left\{\frac{m(\alpha/k)^{k-2}}{n_k},\ m\right\}\right)$$

queries in expectation.

1.2 Related Work. Clique counting, and the special case of triangle counting, have received significant attention in a variety of models. We refer the interested reader to related work sections of [20] and [24] for

[4] A family of graphs is said to be minor-closed if it is closed under vertex removals, edge removals and edge contractions.

general references. We will focus on algorithms for low arboricity graphs.

The starting point for such algorithms is the seminal work of Chiba and Nishizeki, who give an $O(n + m\alpha^{k-2})$ algorithm for enumerating k-cliques in a graph of arboricity at most α [10]. The usual approach to exploit the arboricity is to use degree or degeneracy orientations, and this method has appeared in a number of theoretical and practical results on triangle and clique counting [12, 56, 7, 30, 39, 16]. Recent work by Kopelowitz et al. shows that improving the $O(m\alpha)$ bound for triangle counting is 3-SUM hard [41].

Our work follows a line of work on estimating subgraph counts using sublinear algorithms. The first results were average degree estimation results of Feige [29] and Goldreich and Ron [34]. These ideas were extended by Gonen et al. to estimate star counts [35]. This was the first paper that looked at the problem of estimating triangles, albeit from a lower bound perspective. Eden et al. gave the first sublinear algorithm for approximating the number of triangles. Their result was generalized by the authors for k-clique counting (as mentioned earlier) [24]. Recently, Assadi et al. [2] gave an algorithm for approximately counting the number of occurrences of any arbitrary graph H in an input graph G, denoted by n_H, in $O(m^{\rho(H)}/n_H)$ time, where $\rho(H)$ is the fractional edge cover of H.[5] Their algorithm works in a strictly more powerful model that also allows for uniform-edge queries and was previously studied in the context of sublinear algorithms by Aliakbarpour et al. [1].

The relevance of arboricity for sublinear algorithms was discovered in the context of estimating stars (or degree moments) in previous work by the authors [22]. In that work, standard lower bounds for estimating degree moments could be avoided for low arboricity graphs, just as in Theorem 1.1. Recent work of Eden et al. gives a sublinear (bicriteria) algorithm for property testing arboricity [21].

On the data mining side, Dasgupta et al. and Chierichetti et al. consider sublinear algorithms for estimating the average degree, in weaker models than the standard property testing model [17, 11]. These results require extra assumptions on the graphs. Eden et al. build on the ideas developed in work mentioned earlier to get a practical algorithm for estimating the degree distribution [19].

There is a rich literature on sublinear algorithms for estimating other graph parameters such as the minimum spanning tree, matchings, and vertex covers [9, 15, 14, 50, 61, 52, 50, 42, 61, 36, 51].

1.3 Organization of the paper. Our algorithm and its analysis are quite involved. In Section 2 we give a fairly elaborate (but informal) overview of our algorithm and the ideas behind it. After introducing some preliminaries and defining some central notions (in Sections 3 and 4), we provide our algorithm and the main procedures it uses (in Sections 5 and 6). The full details of its analysis, as well as the proofs of the lower bound are given in the full version of this paper [23], which from here on we refer to as *the full version*.

2 Overview of the algorithm and lower bound

We start with describing the main ideas behind the algorithm. As we explain below, our starting point is similar to the one applied in [24] for approximately counting the number of k-cliques in general graphs (and that of [20], for $k = 3$). However, in order to exploit the fact that the graphs we consider have bounded arboricity, we depart quite early from the [24] algorithm, and introduce a variety of new ideas. For the sake of simplicity of the presentation, assume that $\alpha < n_k^{1/k}$ and that ε is a constant, so that we aim for an upper bound of roughly $O(n\alpha^{k-1}/n_k)$ (recall that $m \leq \alpha n$). In what follows we refer to [24] as ERS.

2.1 Common starting point with ERS and the arboricity challenge. Assume we uniquely and arbitrarily assign each k-clique to one of its vertices. For a vertex v let $w(v)$ denote the number of k-cliques assigned to it, where we refer to this value as the *weight* of v. Consider sampling a set $\mathcal{R}$ of vertices uniformly at random,[6] and let $w(\mathcal{R}) = \sum_{v\in\mathcal{R}} w(v)$. Clearly, $\mathrm{Ex}[w(\mathcal{R})] = \frac{n_k}{n} \cdot |\mathcal{R}|$. However, $w(\mathcal{R})$ might have a large variance. For example, consider the case of $k = 3$ and the wheel graph, where it is possible that the central vertex is assigned all the triangles. Hence, we need an assignment rule that assigns almost all k-cliques, but minimizes the number of k-cliques assigned to any vertex. Furthermore, the rule should be efficiently computable. That is, given a vertex v and a k-clique C, it should be easy to verify whether C is assigned to v. Assume for now that we have such an assignment rule, and that $w(\mathcal{R})$ is indeed close to its expected value.

The next step is to estimate $w(\mathcal{R})$. Let $E_{\mathcal{R}}$ denote the set of edges incident to the vertices of $\mathcal{R}$, and assume that $|E_{\mathcal{R}}|$ is close to its expected value $\frac{m}{n} \cdot |\mathcal{R}|$.

[5] The fractional edge cover of a graph $H = (V_H, E_H)$ is a mapping $\psi : E_H \to [0, 1]$ such that for each vertex $a \in V_H$, $\sum_{e\in E_H, a\in e} \psi(e) \geq 1$. The fractional edge-cover number $\rho(H)$ of H is the minimum value of $\sum_{e\in E_H} \psi(e)$ among all fractional edge covers ψ.

[6] The algorithm may actually obtain a multiset, but in this exposition, we abuse terminology and call it a 'set'.

In ERS, $w(\mathcal{R})$ is approximated by sampling uniform edges in $E_{\mathcal{R}}$ and extending them to k-cliques. Consider first the (easy) case where all the vertices have degree $O(\sqrt{m})$. In this case it is possible to extend an edge (u, v) for $u \in \mathcal{R}$ to a (potential) k-clique by sampling $k-2$ neighbors of u, each with probability roughly $1/\sqrt{m}$ (and checking whether we obtained a clique). The probability that this process yields a k-clique is roughly $\frac{w(\mathcal{R})}{|E_{\mathcal{R}}|\cdot\sqrt{m}^{k-2}} \approx \frac{n_k}{m^{k/2}}$. By repeating the above process $O(m^{k/2}/n_k)$ times,[7] it is possible to can get an estimate of $w(\mathcal{R})$ and thus of n_k (assuming an efficient verification procedure for the assignment rule). For the case when degrees are much larger than $\sqrt{m}$, ERS gives a more complex procedure that extends edges to (potential) k-cliques. In the end, each k-clique is still sampled with probability roughly $n_k/m^{k/2}$.

In our setting (where the arboricity is at most α) the simple scenario discussed above of vertex degrees bounded by $O(\sqrt{m})$ corresponds to the case that all vertex degrees are $O(\alpha)$. In this special case we can extend an edge to a (potential) k-clique in the same manner as ERS, and get that the success probability of sampling a k-clique is $\Omega\left(\frac{n_k}{m\alpha^{k-2}}\right)$. Unfortunately, it is not clear how to adapt the ERS approach for the unbounded-degrees case and obtain a dependence on α instead of $\sqrt{m}$.

To give a better sense of the challenge, we focus on approximating the number of triangles (i.e., n_3) and consider a graph G with $m = \Theta(n)$ and $n_3 = \Theta(\sqrt{n})$. The upper bound of ERS allows for a "budget" of $O^*\left(\frac{n}{n_3^{1/3}} + \frac{m^{3/2}}{n_3}\right) = O^*(n)$ queries. That is, since $m = \Theta(n)$, they can essentially "afford" to read the entire graph. Now assume that we are also given that $\alpha = O(1)$. Our upper bound only allows for $O^*\left(\min\left\{\frac{n\alpha^2}{n_3}, \frac{n}{n_3^{1/3}} + \frac{m\alpha}{n_3}\right\}\right) = O^*(\sqrt{n})$ queries, that is, a strictly sublinear number of queries. Recall that bounded arboricity does not imply bounded vertex degrees. Our main challenge is to exploit the bounded arboricity so as to deal with vertices with high degrees and with high variance in the number of triangles that reside on different vertices (and edges).

Therefore, at this point, we depart from the approach of ERS.

[7] The observant reader may be worried that this requires knowing m and n_k, where the former is not provided to the algorithm and the latter is just what we want to estimate. However, constant factor estimates of both suffice for our purposes. For m this can be obtained using [22], and for n_k this assumption can be removed by performing a geometric search. For details see the full version.

2.2 An iterative sampling process. The ERS algorithm can be viewed as a three-step process. It first samples vertices, then samples edges (incident to the sampled vertices), and then (in one step) samples k-cliques that are extensions of these edges. To get a complexity depending on the arboricity, we devise an iterative clique sampling process. In iteration t, we obtain a sample of t-cliques, based on the sample of $(t-1)$-cliques from the previous iteration.

It is crucial in our analysis to distinguish *ordered* cliques from unordered cliques. An unordered t-clique T is a set of t vertices $T = \{v_1, \ldots, v_t\}$ (such that every two vertices are connected), while an ordered t-clique is a tuple of t vertices $\vec{T} = (v_1, \ldots, v_t)$ such that $\{v_1, \ldots, v_t\}$ is a clique. We say that $\vec{T} = (v_1, \ldots, v_t)$ *participates* in a clique C, if $\{v_1, \ldots, v_t\} \subseteq C$. We also extend the (yet undefined) assignment rule to allow assigning k-cliques to ordered t-cliques for any $t \leq k$ (and not just to vertices, which is the special case of $t = 1$). For an ordered t-clique $\vec{T}$, let $w(\vec{T})$ be the number of k-cliques that are assigned to $\vec{T}$, and for a set of ordered t-cliques $\mathcal{R}$, let $w(\mathcal{R}) = \sum_{\vec{T}\in\mathcal{R}} w(\vec{T})$. Our goal is to estimate $w(V) \approx n_k$. We defer the discussion of the assignment rule and for now focus on the algorithm.

The algorithm starts by sampling a set of s_1 ordered 1-cliques (vertices), denoted $\mathcal{R}_1$. Assume that $w(\mathcal{R}_1) \approx \frac{n_k}{n} \cdot s_1$. The algorithm next samples a set of s_2 ordered 2-cliques (ordered edges), denoted $\mathcal{R}_2$, incident to the vertices of $\mathcal{R}_1$. For $t > 2$, the t^{th} iteration extends $\mathcal{R}_t$ to $\mathcal{R}_{t+1}$, as described next.

For an ordered t-clique $\vec{T}$, let $d(\vec{T})$ be the degree of the *minimum-degree* vertex in $\vec{T}$, and for a set of ordered cliques $\mathcal{R}$, let $d(\mathcal{R}) = \sum_{\vec{T}\in\mathcal{R}} d(\vec{T})$. The sampling of the set $\mathcal{R}_{t+1}$ is done by repeating the following s_{t+1} times: sample a clique $\vec{T}$ in $\mathcal{R}_t$ with probability proportional to $d(\vec{T})/d(\mathcal{R}_t)$ and then select a uniform neighbor of the least degree vertex in $\vec{T}$. Hence, each $(t+1)$-tuple that is an extension of an ordered t-clique in $\mathcal{R}_t$ is sampled with probability $\frac{d(\vec{T})}{d(\mathcal{R}_t)} \cdot \frac{1}{d(\vec{T})} = \frac{1}{d(\mathcal{R}_t)}$. For each sampled $(t+1)$-tuple, the algorithm checks whether it is a $(t+1)$-clique, and if so, adds it to $\mathcal{R}_{t+1}$. Suppose that the weight function (defined by the assignment rule) has the following property. The weight $w(\mathcal{R}_t)$ is the sum of the weights taken over all ordered $(t+1)$-cliques that are extensions of the ordered t-cliques in $\mathcal{R}_t$. We can conclude that the expected value of $w(\mathcal{R}_{t+1})$ is $\frac{w(\mathcal{R}_t)}{d(\mathcal{R}_t)} \cdot s_{t+1}$.

We need to get good upper bounds for s_{t+1}, while ensuring that $w(\mathcal{R}_{t+1})$ is concentrated around its mean. Note that the probability of getting a $(t+1)$-clique is inversely proportional to $d(\mathcal{R}_t)$. Thus, we need good

upper bounds on this quantity, to upper bound s_{t+1}. This is where the arboricity enters the picture. Let $\mathcal{C}_t$ denote the set of t-cliques in the graph. We give a simple argument proving that $d(\mathcal{C}_t) = \sum_{\vec{T} \in \mathcal{C}_t} d(\vec{T}) = O(m\alpha^{t-1})$. (Note that the case $t = 2$ is precisely the Chiba and Nishizeki bound $\sum_{(u,v)\in E} \min\{d(u), d(v)\} = O(m\alpha)$ [10].) We then show that $d(\mathcal{R}_t)$ is bounded as a function of $d(\mathcal{C}_t)$.

2.3 Desired properties of the assignment rule. Recall that we need to ensure that, with high probability, $w(\mathcal{R}_1)$ is close to its expected value, which should be close to $\frac{n_k}{n} \cdot s_1$, and that for every $t \geq 1$, $w(\mathcal{R}_{t+1})$ is close to $\frac{w(\mathcal{R}_t)}{d(\mathcal{R}_t)} \cdot s_{t+1}$. In addition, we need to efficiently verify the assignment rule. We achieve this by defining an assignment rule that has the following properties.

1. $w(V) \approx n_k$. This ensures that the expected value of $w(\mathcal{R}_1)$ is approximately $\frac{n_k}{n} \cdot s_1$.
2. For every t, the sum of the weights taken over all ordered $(t+1)$-cliques that are extensions of the ordered t-cliques in $\mathcal{R}_t$ equals $w(\mathcal{R}_t)$. This ensures that for every t, $\mathrm{Ex}[w(\mathcal{R}_{t+1})] = \frac{w(\mathcal{R}_t)}{d(\mathcal{R}_t)} \cdot s_{t+1}$.
3. For every ordered t-clique $\vec{T}$, $w(\vec{T})$ is not too large. This ensures that with high probability $w(\mathcal{R}_{t+1})$ is close to its expected value for all t, for a sufficiently large sample size s_{t+1} (which depends on this upper bound on $w(\vec{T})$ as well as on $d(\mathcal{R}_t)$).
4. Given a k-clique C and an ordered t-clique $\vec{T} = (v_1, \ldots, v_t)$ such that $\{v_1, \ldots, v_t\} \subseteq C$, we can efficiently determine if C is assigned to $\vec{T}$. This ensures that when we get the final set $\mathcal{R}_k$ of ordered k-cliques, we can compute its weight (and deduce an estimate of n_k).

We introduce key notions in the definition of such an assignment rule.

2.4 Sociable cliques and the assignment rule. For an ordered t-clique $\vec{T}$, let $c_k(\vec{T})$ denote the number of k-cliques containing $\vec{T}$. An ordered t-clique $\vec{T}$ is called *sociable* if $c_k(\vec{T})$ is above a threshold $\tau_t \approx \alpha^{t-1}$. Otherwise, the clique is called *non-sociable*. For a k-clique $C = \{v_1, \ldots, v_k\}$, let $\mathcal{O}(C)$ be the set of all ordered k-cliques corresponding to the $k!$ tuples inducing C. Let $\mathcal{O}'(C)$ be the subset of $\mathcal{O}(C)$ that contains ordered k-cliques in $\mathcal{O}(C)$ such that *all prefixes are non-sociable*. Consider the assignment rule that assigns C to the first (in lexicographic order) $\vec{C} \in \mathcal{O}'(C)$ and to each of its prefixes.

In a central lemma we prove that the number of k-cliques that are not assigned by this assignment rule to any ordered k-clique (and its prefixes) is relatively small. The proof relies on the sociability thresholds $\{\tau_t\}$ and the fact that the graph has arboricity at most α. We note that ERS also defined the notion of sociable vertices (as vertices that participate in too many k-cliques). However, their argument for bounding the number of unassigned k-cliques was simpler, as they did not define and account for sociable cliques for $t > 1$.

The aforementioned assignment rule addresses Properties 1 to 3. We are left with Property 4 (and how it fits in the big picture).

2.5 Verifying an assignment and costly cliques. Recall that in the last iteration of the algorithm, it has a set $\mathcal{R}_k$ of ordered k-cliques, and it needs to compute $w(\mathcal{R}_k)$ (which can be translated to an estimate of n_k). Namely, for each ordered k-clique $\vec{C} = (v_1, \ldots, v_k)$ in $\mathcal{R}_k$, the algorithm needs to verify whether the corresponding k-clique $C = \{v_1, \ldots, v_k\}$ is assigned to $\vec{C}$. This requires to verify whether $\vec{C}$, and each of its prefixes, is non-sociable. Furthermore, it requires verifying that $\vec{C}$ is the first such ordered k-clique (in $\mathcal{O}(C)$).

For an ordered t-clique $\vec{T}$, consider the subgraph $G_{\vec{T}}$ induced by the set of vertices that neighbor every vertex in $\vec{T}$. Observe that $c_k(\vec{T})$ equals the number of $(k-t)$-cliques in $G_{\vec{T}}$. Therefore, deciding whether $\vec{T}$ is sociable amounts to deciding whether the number of $(k-t)$-cliques in the subgraph $G_{\vec{T}}$ is greater than τ_t. Indeed this is similar to our original problem of estimating the number of $(k-t)$-cliques in a graph, except that it is applied to a subgraph $G_{\vec{T}}$ of our original graph G. Unfortunately, we do not have direct query access to such subgraphs. To illustrate this, consider the case of $t = 1$ so that $\vec{T}$ consists of single vertex v. While we can sample uniform vertices in the subgraph $G_{(v)}$, we cannot directly perform neighbor queries (without incurring a possibly large cost when simulating queries to $G_{(v)}$ by performing queries to G).

However, we show that we can still follow the high-level structure of our iterative sampling algorithm (though there are a few obstacles). Specifically, we initialize $\mathcal{R}_t = \{\vec{T}\}$, and for each $j = t, \ldots, k-1$, we sample a set of ordered $(j+1)$-cliques $\mathcal{R}_{j+1}$ given a set of ordered j-cliques $\mathcal{R}_j$, exactly as described in Section 2.2. The first difficulty that we encounter is the following. The success probability of sampling an ordered $(j+1)$-clique that extends an ordered j-clique in $\mathcal{R}_j$ is inversely proportional to $d(\mathcal{R}_j)$. Unfortunately, here we cannot argue that with high probability $d(\mathcal{R}_j)$ can be upper bounded as a function of $d(\mathcal{C}_j)$ (which is $O(m\alpha^{j-1})$). The reason is that while the algorithm described in Section 2.2 starts with a uniform sample of vertices $\mathcal{R}_1$ (that the following samples $\mathcal{R}_j$ build on),

here we start with $\mathcal{R}_t = \{\vec{T}\}$ for an arbitrary t-clique $\vec{T}$.

We overcome this obstacle by defining the notion of *costly* cliques. We say that an ordered t-clique $\vec{T}$ is costly if for some $j \geq t$, $d(\mathcal{C}_j(\vec{T}))$ is too large, where $\mathcal{C}_j(\vec{T})$ is the set of j-cliques that $\vec{T}$ participates in. For such ordered t-cliques, we cannot efficiently verify whether they are sociable. Thus, we modify our assignment rule so that costly cliques are not assigned any k-clique (even if they are non-sociable). We prove that the additional loss in unassigned k-cliques is small and that we can efficiently determine if an ordered t-clique is costly. So we start with $\mathcal{R}_t = \{\vec{T}\}$, apply the iterative process, and obtain a set of ordered k-cliques $\mathcal{R}_k$ (that are all extensions of $\vec{T}$). To determine if $\vec{T}$ is sociable, we need to estimate $c_k(\vec{T})$, i.e., the number of $(k-t)$ cliques in $G_{\vec{T}}$. Luckily, it suffices to make this decision approximately. For the analysis to go through, it suffices to distinguish between the case that $c_k(\vec{T})$ is "too large", and the case that it is "sufficiently small". Therefore, given $\mathcal{R}_k$, the final decision (regarding the sociability of $\vec{T}$) can be made just based on $|\mathcal{R}_k|$.

2.6 Summary of our main new ideas and where arboricity comes into play. The following are the main differences and new ideas as compared to ERS, with an emphasis on the role of bounded arboricity.

1. We introduce an iterative sampling process that, starting from a uniform sample $\mathcal{R}_1$ of vertices, creates intermediate samples $\mathcal{R}_t$ of ordered t-cliques, until it obtains a sample of ordered k-cliques. Arboricity comes into play here since the probability of obtaining an ordered $(t+1)$-clique that can be added to $\mathcal{R}_{t+1}$, is inversely proportional to $1/d(\mathcal{R}_t)$, which in turn can be bounded as a function of α (and m).

2. We introduce an assignment rule and corresponding weight function w that ensures two properties. (1) Almost every k-clique is assigned (to some ordered k-clique and all its prefixes), and (2) no ordered clique is assigned too many k-cliques. The former implies that $w(V) \approx n_k$. The latter implies that, in the iterative sampling process, each sample of larger ordered cliques "maintains the weight" (up to an appropriate normalization) of the previous sample.
The arboricity α determines the sociability thresholds $\{\tau_t\}$ (above which an ordered clique is not assigned any k-clique). These thresholds are carefully chosen to ensure that in graphs with arboricity at most α, the number of unassigned k-cliques is sufficiently small. These parameters directly affect the time complexity of the algorithm.

3. We show how the assignment rule can be verified. This translates to determining whether certain ordered cliques are sociable. A key notion is that of costly cliques, whose sociability cannot be determined efficiently. Arboricity also plays a role in their definition and in the proof that the additional loss incurred by not assigning k-cliques to costly ordered cliques is small.

2.7 The lower bound. Both terms in the lower bound are direct generalizations of the lower bound for approximately counting k-cliques in general graphs, first proven in [24] and later simplified by Eden and Rosenbaum [25].

For the case that $n_k \leq \binom{\alpha}{k}$, the proof of the $\Omega\left(\frac{n}{k \cdot n_k^{1/k}}\right)$ term has already appeared in the works mentioned above. For the case that $n_k > \binom{\alpha}{k}$, the proof is based on a simple hitting argument as follows. We start with a fixed graph G' with n vertices, m edges, arboricity α, and no k-cliques. Now, let G_1 consist of $r = n_k/\binom{\alpha}{k}$ disjoint α-cliques, with a disjoint copy of G'. Let G_2 consist of $r\alpha$ isolated vertices and a disjoint copy of G'. Hence, in both graphs G_1 and G_2 there are $\Theta(n)$ vertices, $\Theta(m)$ edges, and arboricity α. Furthermore, in both graphs there are no edges between the different subgraphs, and we consider a random labeling of the vertex names. Clearly, an algorithm cannot distinguish between the two graphs, unless it hits one of the cliques, which happens with probability $\frac{r \cdot \alpha}{n} = \Theta\left(\frac{k^k \cdot n_k}{n \cdot \alpha^{k-1}}\right)$. Therefore, any algorithm for estimating the number of k-cliques must perform $\Omega\left(\frac{n\alpha^{k-1}}{k^k \cdot n_k}\right)$ queries in expectation.

The proof of the $\Omega\left(\min\{\frac{m(\alpha/k)^{k-2}}{n_k}, m\}\right)$ term is more involved, and is based on a reduction from a generalization of the set-disjointness communication complexity problem.

3 Preliminaries

For an integer j, the set $\{1, \ldots, j\}$ is denoted by $[j]$. For a pair of integers $i \leq j$, the set of integers $\{i, \ldots, j\}$ is denoted by $[i, j]$. For a multiset S, we use $|S|$ to denote the sum of multiplicities of the items in S. Our algorithm gets parameters k and ε, where we assume that $\varepsilon < 1/2k^2$ (or else we set $\varepsilon = 1/2k^2$).

Let $G = (V, E)$ be a graph with n vertices, m edges, and arboricity $\alpha(G)$. As noted in Section 2, we distinguish between a t-clique, which is a set of t vertices $T = \{v_1, \ldots, v_t\}$ (with an edge between every pair of vertices in the set), and an *ordered* t-clique, which is a t-tuple of t distinct vertices $\vec{T} = (v_1, \ldots, v_t)$ such that $\{v_1, \ldots, v_t\}$ is a clique. For an ordered t-clique $\vec{T} = (v_1, \ldots, v_t)$, we use $U(\vec{T})$ to denote the corresponding unordered t-clique $\{v_1, \ldots, v_t\}$. For cliques (ordered

cliques) of size 1, that is, vertices, we may use v instead of $\{v\}$ (respectively, (v)), and similarly for cliques of size 2 (edges). We let $\mathcal{C}_t(G)$ denote the set of t-cliques in G, and $n_t(G) = |\mathcal{C}_t(G)|$. For the set of ordered t-cliques in G we use $\mathcal{O}_t(G)$. When G is clear from the context, we use the shorthand $\mathcal{C}_t$, n_t and $\mathcal{O}_t$, respectively.

DEFINITION 3.1. (CLIQUE'S LEAST DEGREE VERTEX) *For a clique (or ordered clique) C we let $\Gamma(C)$ denote the set of neighbors of C's minimal-degree vertex (breaking ties by ids) and let $d(C) = |\Gamma(C)|$. We refer to $d(C)$ as the* degree of the (ordered) clique *and to $\Gamma(C)$ as its* set of neighbors. *For a set (or multiset) of cliques (or ordered cliques) $\mathcal{R}$, we use the notation $d(\mathcal{R})$ for $\sum_{C\in\mathcal{R}} d(C)$.*

We stress that $\Gamma(C)$ (and respectively, $d(C)$) does not refer to the union of neighbors of vertices in C, but only to the neighbor of a single designated vertex in C.

The proofs of the next two claims appear in the full version.

CLAIM 3.1. *For every t, $d(\mathcal{C}_t(G)) \leq 2m \cdot \alpha(G)^{t-1}$.*

CLAIM 3.2. *For every $t \geq 2$, $n_t(G) \leq \frac{2\alpha(G)}{t} \cdot n_{t-1}(G)$.*

As a corollary of Claim 3.2, we obtain.

COROLLARY 3.1. *For every $1 \leq t < k$,*

$$n_k(G) \leq \frac{t!}{k!} \cdot n_t(G) \cdot (2\alpha(G))^{k-t} .$$

4 Weight functions and assignments

As explained in the overview of our algorithm, a central component in our approach is a weight function defined over ordered cliques. We shall be interested in a weight function that is *legal* in the following sense.

DEFINITION 4.1. (A LEGAL WEIGHT FUNCTION) *A weight function $w : \bigcup_{t=1}^{k} \mathcal{O}_t \to \mathbb{N}$ is* legal *if it satisfies the following.*

1. *For every ordered k-clique $\vec{C}$, $w(\vec{C}) \in \{0, 1\}$, and for every unordered k-clique C, there is at most one ordered k-clique $\vec{C}$ such that $C = U(\vec{C})$ and $w(\vec{C}) = 1$.*
2. *For every $t \in [k-1]$ and for every ordered t-clique $\vec{T}$, $w(\vec{T}) = \sum_{\vec{T}'\in\mathcal{O}_{t+1}(\vec{T})} w(\vec{T}')$.*

For a multiset of ordered cliques $\mathcal{R}$, we let $w(\mathcal{R}) = \sum_{\vec{T}\in\mathcal{R}} w(\vec{T})$.

By the above definition,

FACT 4.1. *Let w be a legal weight function. Then $w(V) \leq n_k$.*

We next show how to define a weight function based on a subset $\mathcal{A} = \bigcup_{t=1}^{k} \mathcal{A}_t$ such that $\mathcal{A}_t \subseteq \mathcal{O}_t$, which we refer to as a subset of *active* ordered cliques. Referring to the notions introduced informally in Section 2, the intention is that active ordered cliques will be non-sociable and non-costly where these notions are formally defined in the in the full version. The weight function is closely linked to the notion of assigning k-cliques to ordered cliques (as becomes clear in Definition 4.3). For now our goal is to define such a weight function that is legal, and such that we can easily verify (based on $\mathcal{A}$) whether an ordered k-clique has weight 1 or 0. We would like to devise a weight function w such that $w(V)$ is not much smaller than n_k, and that the weight of very ordered clique is appropriately bounded. We later provide sufficient conditions on $\mathcal{A}$, which ensure that these properties hold.

In what follows, for a set $\mathcal{R}$ of ordered cliques (in particular of the same size), we say that $\vec{T} \in \mathcal{R}$ is *first* in $\mathcal{R}$ if it is lexicographically first. Also, for an ordered t-clique $\vec{T}$ and $j \leq t$, we use $\vec{T}_{\leq j}$ to denote the ordered j-clique formed by the first j elements in $\vec{T}$.

Since we shall be interested in active ordered cliques such that all of their prefixes are also active, it will be useful to define the notion of fully active cliques.

DEFINITION 4.2. (FULLY-ACTIVE CLIQUES) *Let $\mathcal{A}$ be a subset of ordered cliques. An ordered t-clique $\vec{T}$ is* fully active *with respect to $\mathcal{A}$, if all of its prefixes belong to $\mathcal{A}$. That is, $\vec{T}_{\leq j} \in \mathcal{A}$ for every $j \in [t]$. We denote the subset of t-cliques that are fully active with respect to $\mathcal{A}$ by $\mathcal{F}_t^{\mathcal{A}}$.*

We are now ready to define our assignment rule. Recall that for an ordered k-clique $\vec{C} = (v_1, \ldots, v_k)$, we use $U(\vec{C}) = \{v_1, \ldots, v_k\}$ to denote its corresponding unordered clique. Hence, for an unordered k-clique C, $U^{-1}(C)$ is the set of $k!$ ordered k-cliques $\vec{C}'$ such that $U(\vec{C}') = C$.

DEFINITION 4.3. (ASSIGNMENT AND WEIGHT) *Let $\mathcal{A}$ be a subset of ordered cliques. For each k-clique C, if $U^{-1}(C) \cap \mathcal{F}_k^{\mathcal{A}} \neq \emptyset$, then C is* assigned *(with respect to $\mathcal{A}$) to the first ordered k-clique $\vec{C} \in U^{-1}(C) \cap \mathcal{F}_k^{\mathcal{A}}$, and to each ordered t-clique $\vec{C}_{\leq t}$ for $t \in [k-1]$. Otherwise (if $U^{-1}(C) \cap \mathcal{F}_k^{\mathcal{A}} = \emptyset$), C is* unassigned. *That is, we assign the k-clique C to the first ordered k-clique $\vec{C}$ in $U(C)$ that is fully active and to all of its prefixes, if such an ordered clique $\vec{C}$ exists. Otherwise we do not assign C to any ordered clique.*

For each ordered t-clique $\vec{T}$, we let $w^{\mathcal{A}}(\vec{T})$ denote the number of k-cliques that are assigned to $\vec{T}$, and we refer to $w^{\mathcal{A}}(\vec{T})$ as the weight of $\vec{T}$ *(with respect to $\mathcal{A}$).*

Observe that by Definition 4.3, an ordered t-clique $\vec{T}$ is assigned some k-clique C only if it is the t-prefix of some fully active (with respect to $\mathcal{A}$) ordered k-clique $\vec{C}$. This implies that $\vec{T}$ is active, and hence, only ordered cliques in $\mathcal{A}$ can have non-zero weight.

The next claim follows from Definition 4.3.

CLAIM 4.1. *For any subset $\mathcal{A}$ of ordered cliques, $w^{\mathcal{A}}(\cdot)$ is a legal weight function.*

The following definition encapsulates what we require from the resulting weight function $w^{\mathcal{A}}$.

DEFINITION 4.4. (GOOD ACTIVE SUBSET) *For an approximation parameter ε and a vector of weight thresholds $\vec{\tau} = (\tau_1, \ldots, \tau_k)$, we say that a subset $\mathcal{A}$ of ordered cliques is $(\varepsilon, \vec{\tau})$-*good *if the following two conditions hold:*

1. *For every $t \in [k]$ and for every ordered t-clique $\vec{T}$, $w^{\mathcal{A}}(\vec{T}) \leq \tau_t$.*
2. *$w^{\mathcal{A}}(V) \geq (1 - \varepsilon/2)n_k$.*

*If only the first condition holds, then we say that $\mathcal{A}$ is $\vec{\tau}$-*bounded*.*

As we shall discuss in more detail subsequently, we obtain the first item in Definition 4.4 by ensuring that $\mathcal{A}$ includes only ordered cliques that do not participate in too many k-cliques.

5 An oracle based algorithm

In order to make the presentation more modular, we first present an *oracle-based* algorithm. That is, we assume the algorithm, **Approx-Cliques**, is given access to an oracle $\mathcal{Q}^{\mathcal{A}}$ for a subset of active ordered cliques $\mathcal{A}$: for any given ordered clique $\vec{T}$, the oracle $\mathcal{Q}^{\mathcal{A}}$ returns whether $\vec{T} \in \mathcal{A}$. The algorithm also receives an approximation parameter ε, a confidence parameter δ, a "guess estimate" $\widetilde{n}_k$ of n_k, an estimate $\widetilde{m}$ of m, and a vector of weight-thresholds $\vec{\tau}$. Our main claim will roughly be that if $\mathcal{A}$ is $(\varepsilon, \vec{\tau})$-good (as defined in Definition 4.4), $\widetilde{m} \geq m/2$ and $\widetilde{n}_k \leq n_k$, then with probability at least $1 - \delta$ the algorithm outputs a $(1 \pm \varepsilon)$ approximation of n_k, by approximating $w^{\mathcal{A}}(V)$. A constant-factor estimate $\widetilde{m}$ of m can be obtained by calling the moments-estimation algorithm of [22], which is designed to work for bounded-arboricity graphs (applying it simply to the first moment). We can alleviate the need for the parameter $\widetilde{n}_k$, by relying on the search algorithm of [24] .

The algorithm **Approx-Cliques** starts by selecting a uniform sample of vertices (1-cliques). It then continues iteratively, where at the start of each iteration it has a sample of ordered t-cliques $\mathcal{R}_t$. It sends this sample to the procedure **Sample-a-Set**, which returns a sample of ordered $(t+1)$-cliques $\mathcal{R}_{t+1}$. The ordered cliques in $\mathcal{R}_{t+1}$ are extensions of ordered cliques in $\mathcal{R}_t$. Once the algorithm reaches $t = k$, so that it has a sample of ordered k-cliques, it calls the procedure **Is-Assigned** on each ordered k-clique $\vec{C}$ in $\mathcal{R}_k$ to check whether it is assigned the unordered clique $U(\vec{C})$ (i.e., $w^{\mathcal{A}}(\vec{C}) = 1$). Finally it returns an appropriately normalized version of the total weight of $\mathcal{R}_k$.

For the sake of the exposition, here we provide a slightly simplified version of the algorithm. In particular, we give approximate settings of the variables in the algorithm, and use the notation $\approx$ for these approximate settings. We also removed two steps in which the algorithm aborts, which are used in order to bound the complexity of the algorithm. For full details (as well as a complete analysis) see full version.

The procedure **Sample-a-Set** (invoked in Step 3b of **Approx-Cliques**), is presented next. Given a multiset $\mathcal{R}_t$ of ordered t-cliques, consider all $(t+1)$-tuples that each corresponds to an ordered t-clique $\vec{T}$ in $\mathcal{R}_t$, and a neighbor v of $\vec{T}$. (For the definition of the neighbors of an ordered clique $\vec{T}$ and its degree $d(\vec{T})$ refer to Definition 3.1.) **Sample-a-Set** samples uniformly from these tuples, and includes in $\mathcal{R}_{t+1}$ those $(t+1)$-tuples that are ordered $(t+1)$-cliques. Constructing a data structures that supports sampling each $\vec{T} \in \mathcal{R}_t$ with probability $d(\vec{T})/d(\mathcal{R}_t)$ when given all degrees $d(\vec{T})$ and $d(\mathcal{R}_t)$ can be implemented in linear time in $|\mathcal{R}_t|$ (see e.g., [59, 60, 43] on generating random variables according to a specified discrete distribution).

For a t-clique $T = \{v_1, \ldots, v_t\}$, and $j \geq t$, we let $\mathcal{C}_j(T)$ denote the set of j-cliques that T participates in. That is, the set of j-cliques T' such that $T \subseteq T'$. For an ordered t-clique $\vec{T}$ we use $\mathcal{C}_j(\vec{T})$ as a shorthand for $\mathcal{C}_j(U(\vec{T}))$, and also say that $\vec{T}$ participates in each $\vec{T}' \in \mathcal{C}_j(\vec{T})$. We let $\mathcal{O}_j(\vec{T})$ denote the set of ordered j-cliques that are extensions of $\vec{T}$. For a multiset of (ordered) t-cliques $\mathcal{R}$, we use $\mathcal{C}_j(\mathcal{R})$ to denote the union of $\mathcal{C}_j(T)$ taken over all $\vec{T} \in \mathcal{R}$, where here in "union" we mean with multiplicity,[8] and $\mathcal{O}_j(\mathcal{R})$ is defined analogously. We extend the definitions of $\mathcal{C}_j$ and $\mathcal{O}_j$ to t-tuples such that for a t-tuple $\vec{T}$ that does not correspond to a t-clique, $\mathcal{C}_j(\vec{T})$ and $\mathcal{O}_j(\vec{T})$ are mapped to the empty set. Finally, for an ordered t-clique $\vec{T} = (v_1, \ldots, v_t)$ and a vertex u, we use $(\vec{T}, u)$ as a shorthand for the $(t+1)$-tuple $(v_1, \ldots, v_t, u)$.

It can be proved for an appropriate setting of the parameter s_{t+1}, the set $\mathcal{R}_{t+1}$ returned by the procedure is "typical" with respect to the set $\mathcal{R}_t$. Essentially we prove that with high probability, $w(\mathcal{R}_{t+1})$ is close to its

[8] The formal term should be "sum", but since "sum" is usually used in the context of numbers, we prefer to use "union".

Approx-Cliques$(n, k, \alpha, \varepsilon, \delta, \widetilde{m}, \widetilde{n}_k, \vec{\tau}, \mathcal{Q}^{\mathcal{A}})$

1. Define $\mathcal{R}_0 = V$, $d(\mathcal{R}_0) = n$ and set $\widetilde{w}_0 \leftarrows \widetilde{n}_k$.
2. Sample $s_1 \leftarrows \frac{n\tau_1}{\widetilde{n}_k}$ vertices u.a.r. and let $\mathcal{R}_1$ be the chosen multiset.
3. For $t = 1$ to $k-1$ do:
   (a) Compute $d(\mathcal{R}_t)$ and set $\widetilde{w}_t \leftarrows \frac{\widetilde{w}_{t-1}}{d(\mathcal{R}_{t-1})} \cdot s_t$ and $s_{t+1} \leftarrows \frac{d(\mathcal{R}_t)\cdot\tau_{t+1}}{\widetilde{w}_t}$.
   (b) Invoke **Sample-a-Set**$(t, \mathcal{R}_t, s_{t+1})$ and let $\mathcal{R}_{t+1}$ be the returned multiset.
4. Let $\widehat{n}_k = \frac{n\cdot d(\mathcal{R}_1)\cdot\ldots\cdot d(\mathcal{R}_{k-1})}{s_1\cdot\ldots\cdot s_k} \cdot \sum_{\vec{C}\in\mathcal{R}_k}$ **Is-Assigned**$(\vec{C}, \mathcal{Q}^{\mathcal{A}})$.
5. **Return** $\widehat{n}_k$.

Sample-a-Set$(t, \mathcal{R}_t, s_{t+1})$

1. Compute $d(\mathcal{R}_t)$ and set up a data structure to sample each $\vec{T} \in \mathcal{R}_t$ with probability $d(\vec{T})/d(\mathcal{R}_t)$.
2. Initialize $\mathcal{R}_{t+1} = \emptyset$.
3. For $\ell = 1$ to s_{t+1}:
   (a) Invoke the data structure to generate an ordered clique $\vec{T}_\ell$.
   (b) Query degrees of vertices in $\vec{T}_\ell$, and find a minimum degree vertex $u \in \vec{T}_\ell$.
   (c) Sample a random neighbor v_ℓ of u.
   (d) If $(\vec{T}_\ell, v_\ell)$ is an ordered $(t+1)$-clique, add it to $\mathcal{R}_{t+1}$.
4. **Return** $\mathcal{R}_{t+1}$.

expected value, $\frac{w(\mathcal{R}_t)}{d(\mathcal{R}_t)}\cdot s_{t+1}$, and that $d(\mathcal{C}_j(\mathcal{R}_{t+1}))$ is not much larger than its expected value $\frac{d(\mathcal{C}_j(\mathcal{R}_t))}{d(\mathcal{R}_t)}\cdot s_{t+1}$.

The procedure **Is-Assigned** (invoked in Step 4 of **Approx-Cliques**) decides whether a given ordered k-clique $\vec{C}$ is assigned $U(\vec{C})$, i.e., whether $w^{\mathcal{A}}(\vec{C}) = 1$. This is done following Definition 4.3, given access to an oracle for $\mathcal{A}$. In Section 6 we replace the oracle by an explicitly defined procedure.

As stated at the start of this section, we show that if $\mathcal{Q}^{\mathcal{A}}$ is an oracle for an $(\varepsilon, \vec{\tau})$-good subset $\mathcal{A}$, $\widetilde{m} \geq m/2$ and $\widetilde{n}_k \leq n_k$, then with probability at least $1-\delta$, **Approx-Cliques** returns an estimate $\widehat{n}_k$ such that $\widehat{n}_k \in (1\pm\varepsilon)n_k$. The precise statement regarding the correctness and complexity of **Approx-Cliques** as well as its proof, are provided in the full version.

6 Implementing an oracle for good subset $\mathcal{A}$

In this section we describe a (randomized) procedure named **Is-Active**, that implements an oracle for a subset $\mathcal{A}$, where with high probability, $\mathcal{A}$ is $(\varepsilon, \vec{\tau})$-good for an appropriate setting of $\vec{\tau}$. For an ordered t-clique $\vec{T}$, recall that $c_k(\vec{T})$ denotes the number of k-cliques that $\vec{T}$ participates in (that is, $c_k(\vec{T}) = |\mathcal{C}_k(\vec{T})|$). The procedure aims at determining whether $c_k(\vec{T})$ is larger than τ_t, in which case it is not included in $\mathcal{A}$. This ensures that the first item in Definition 4.4 holds, since $w^{\mathcal{A}}(\vec{T}) \leq c_k(\vec{T}) \leq \tau_t$. On the other hand, the setting of $\vec{\tau}$ is such that despite not including in $\mathcal{A}$ all ordered cliques $\vec{T}$ for which $c_k(\vec{T})$ is above the allowed threshold, we can still prove that the second item in Definition 4.4 holds as well.

In order to give the idea behind the procedure **Is-Active**, consider the special case that $t = 1$ and $T = \{v\}$ for some vertex v. Observe that $c_k(v)$ is the number of k-cliques in the subgraph induced by v and its neighbors. Roughly speaking, for $i = 1$, the procedure **Is-Active** works similarly to the algorithm **Approx-Cliques**, subject to setting $\mathcal{R}_1 = \{v\}$. Namely, starting from $t = 1$, and using the procedure **Sample-a-Set**, it iteratively selects a sample $\mathcal{R}_{t+1}$ of ordered $(t+1)$-cliques, given a sample $\mathcal{R}_t$ of ordered t-cliques. Once it obtains $\mathcal{R}_k$ it uses $|\mathcal{R}_k|$ to decide whether $c_k(v)$ is too large (and hence should not be included in $\mathcal{A}$). As opposed to **Approx-Cliques**, here we do not ask for a precise estimate of $c_k(v)$, and hence this decision is simpler. In addition, the invariant that we would like to maintain is that the average number of k-cliques that ordered cliques in $\mathcal{R}_{t+1}$ participate in, does not deviate by much from the average number for $\mathcal{R}_t$. The procedure generalizes to $i > 1$ by setting $\mathcal{R}_i = \{\vec{I}\}$ and starting the sampling process with $t = i$.

Observe that the procedure **Is-Active** may exit (in Step 2c) before reaching $t = k-1$ if s_{t+1} is above a certain threshold. This early exit (with an output of Non-Active) addresses the case that $\vec{I}$ is "costly" (this was informally defined Section 2.5).

As in the case of the algorithm **Approx-Cliques**, here we give a slightly simplified version of **Is-Active**.

By replacing the calls to an oracle $\mathcal{Q}^{\mathcal{A}}$ in the algorithm **Approx-Cliques** with calls to **Is-Active**, we obtain an algorithm that with high probability

Is-Assigned$(\vec{C}, \mathcal{Q}^{\mathcal{A}})$

1. Let $C = U(\vec{C})$.
2. For each $\vec{C}' \in U^{-1}(C)$, check whether $\vec{C}'$ is fully active with respect to $\mathcal{A}$, i.e., whether $\vec{C} \in \mathcal{F}_k^{\mathcal{A}}$, by calling $\mathcal{Q}^{\mathcal{A}}$ on every prefix $\vec{C}'_{\leq t}$ for $t \in [k-1]$.
3. If $\vec{C} \in \mathcal{F}_k^{\mathcal{A}}$ and it is the first ordered k-clique in $\mathcal{F}_k^{\mathcal{A}}$, then **return** 1, otherwise **return** 0.

Is-Active$(i, \vec{I}, k, \alpha, \varepsilon, \delta, \widetilde{n}_k, \widetilde{m}, \vec{\tau})$

1. Let $\mathcal{R}_i = \left\{\vec{I}\right\}$ and $\widetilde{w}_i \eqsim \tau_i$.
2. For $t = i$ to $k-1$ do:
   (a) Compute $d(\mathcal{R}_t)$.
   (b) For $t > i$ set $\widetilde{w}_t \eqsim \frac{\widetilde{w}_{t-1}}{d(\mathcal{R}_{t-1})} \cdot s_t$ and $s_{t+1} \eqsim \frac{d(\mathcal{R}_t) \cdot \tau_{t+1}}{\widetilde{w}_t}$.
   (c) If s_{t+1} is larger than $\frac{\widetilde{m}\alpha^{t-1} \cdot \tau_{t+1}}{\widetilde{n}_k}$, then **return** Non-Active.
   (d) Invoke **Sample-a-Set**$(t, \mathcal{R}_t, s_{t+1})$ and let $\mathcal{R}_{t+1}$ be the returned multiset.
3. Set $\widehat{c}_k(\vec{I}) := \frac{d(\mathcal{R}_i) \cdot \ldots \cdot d(\mathcal{R}_{k-1})}{s_{i+1} \cdot \ldots \cdot s_k} \cdot |\mathcal{R}_k|$.
4. If $\widehat{c}_k(\vec{I}) \leq \tau_i/4$ then **return** Non-Active. Otherwise, **return** Active.

computes a $(1 \pm \varepsilon)$-estimate of n_k (conditioned on $\widetilde{m} \geq m/2$ and $\widetilde{n}_k < n_k$, where both assumptions can be removed). For full details, see the full version.

References

[1] M. Aliakbarpour, A. S. Biswas, T. Gouleakis, J. Peebles, R. Rubinfeld, and A. Yodpinyanee. Sublinear-time algorithms for counting star subgraphs via edge sampling. *Algorithmica*, pages 1–30, 2017.

[2] S. Assadi, M. Kapralov, and S. Khanna. A simple sublinear-time algorithm for counting arbitrary subgraphs via edge sampling. In *ITCS*, volume 124 of *LIPIcs*, pages 6:1–6:20. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2019.

[3] A. L. Barabási and R. Albert. Emergence of scaling in random networks. *science*, 286(5439):509–512, 1999.

[4] R. Bauer, M. Krug, and D. Wagner. Enumerating and generating labeled k-degenerate graphs. In *Proceedings of the Meeting on Algorithm Engineering & Expermiments*, pages 90–98. Society for Industrial and Applied Mathematics, 2010.

[5] M. Baur, M. Gaertler, R. Görke, M. Krug, and D. Wagner. Generating graphs with predefined k-core structure. In *Proceedings of the European Conference of Complex Systems*. Citeseer, 2007.

[6] L. Becchetti, P. Boldi, C. Castillo, and A. Gionis. Efficient semi-streaming algorithms for local triangle counting in massive graphs. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 16–24, 2008.

[7] J. Berry, L. Fostvedt, D. Nordman, C. Phillips, C. Seshadhri, and A. Wilson. Why do simple algorithms for triangle enumeration work in the real world? In *Innovations in Theoretical Computer Science (ITCS)*, pages 225–234, 2014.

[8] R. S. Burt. Structural holes and good ideas. *American Journal of Sociology*, 110(2):349–399, 2004.

[9] B. Chazelle, R. Rubinfeld, and L. Trevisan. Approximating the minimum spanning tree weight in sublinear time. *SIAM Journal on Computing*, 34(6):1370–1379, 2005.

[10] N. Chiba and T. Nishizeki. Arboricity and subgraph listing algorithms. *SIAM Journal on Computing*, 14(1):210–223, 1985.

[11] F. Chierichetti, A. Dasgupta, R. Kumar, S. Lattanzi, and T. Sarlos. On sampling nodes in a network. In *Conference on the World Wide Web (WWW)*, pages 471–481, 2016.

[12] J. Cohen. Graph twiddling in a MapReduce world. *Computing in Science & Engineering*, 11:29–41, 2009.

[13] J. S. Coleman. Social capital in the creation of human capital. *American Journal of Sociology*, 94:S95–S120, 1988.

[14] A. Czumaj, F. Ergün, L. Fortnow, A. Magen, I. Newman, R. Rubinfeld, and C. Sohler. Approximating the weight of the Euclidean minimum spanning tree in sublinear time. *SIAM Journal on Computing*, 35(1):91–109, 2005.

[15] A. Czumaj and C. Sohler. Estimating the weight of metric minimum spanning trees in sublinear time. *SIAM Journal on Computing*, 39(3):904–922, 2009.

[16] M. Danisch, O. D. Balalau, and M. Sozio. Listing k-cliques in sparse real-world graphs. In *Conference on the World Wide Web (WWW)*, pages 589–598, 2018.

[17] A. Dasgupta, R. Kumar, and T. Sarlos. On estimating the average degree. In *Conference on the World Wide Web (WWW)*, pages 795–806. ACM, 2014.

[18] J. P. Eckmann and E. Moses. Curvature of co-links

uncovers hidden thematic layers in the World Wide Web. *Proceedings of the National Academy of Sciences*, 99(9):5825–5829, 2002.

[19] T. Eden, S. Jain, A. Pinar, D. Ron, and C. Seshadhri. Provable and practical approximations for the degree distribution using sublinear graph samples. In *Conference on the World Wide Web (WWW)*, pages 449–458, 2018.

[20] T. Eden, A. Levi, D. Ron, and C Seshadhri. Approximately counting triangles in sublinear time. In *Foundations of Computer Science (FOCS)*, pages 614–633, 2015.

[21] T. Eden, R. Levi, and D. Ron. Testing bounded arboricity. In *Symposium on Discrete Algorithms (SODA)*, pages 2081–2092, 2018.

[22] T. Eden, D. Ron, and C. Seshadhri. Sublinear time estimation of degree distribution moments: The degeneracy connection. In *International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 7:1–7:13, 2017.

[23] T. Eden, D. Ron, and C. Seshadhri. Faster sublinear approximations of k-cliques for low arboricity graphs, 2018.

[24] T. Eden, D. Ron, and C. Seshadhri. On approximating the number of k-cliques in sublinear time. In *Symposium on Theory of Computing (STOC)*, pages 722–734, 2018.

[25] T. Eden and W. Rosenbaum. Lower bounds for approximating graph parameters via communication complexity. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM)*, pages 11:1–11:18, 2018.

[26] F. Eisenbrand and F. Grandoni. On the complexity of fixed parameter clique and dominating set. *Theoretical Computer Science*, 326(1-3):57–67, 2004.

[27] D. Eppstein, M. Löffler, and D. Strash. Listing all maximal cliques in large sparse real-world graphs. *ACM Journal of Experimental Algorithmics*, 18:3–1, 2013.

[28] D. Eppstein and D. Strash. Listing all maximal cliques in large sparse real-world graphs. In *International Symposium on Experimental Algorithms*, pages 364–375. Springer, 2011.

[29] U. Feige. On sums of independent random variables with unbounded variance and estimating the average degree in a graph. *SIAM Journal on Computing*, 35(4):964–984, 2006.

[30] I. Finocchi, M. Finocchi, and E. G. Fusco. Clique counting in mapreduce: Algorithms and experiments. *ACM Journal of Experimental Algorithmics*, 20:1–7, 2015.

[31] B. Foucault Welles, A. Van Devender, and N. Contractor. Is a friend a friend?: Investigating the structure of friendship networks in virtual worlds. In *CHI Extended Abstracts on Human Factors in Computing Systems*, pages 4027–4032, 2010.

[32] G. Goel and J. Gustedt. Bounded arboricity to determine the local structure of sparse graphs. In *International Workshop on Graph-Theoretic Concepts in Computer Science*, pages 159–167. Springer, 2006.

[33] O. Goldreich. *Introduction to Property Testing*. Cambridge University Press, 2017.

[34] O. Goldreich and D. Ron. Approximating average parameters of graphs. *Random Structures and Algorithms*, 32(4):473–493, 2008.

[35] M. Gonen, D. Ron, and Y. Shavitt. Counting stars and other small subgraphs in sublinear-time. *SIAM Journal on Discrete Mathematics*, 25(3):1365–1411, 2011.

[36] A. Hassidim, J. A. Kelner, H. N. Nguyen, and K. Onak. Local graph partitions for approximation and testing. In *Foundations of Computer Science (FOCS)*, pages 22–31, 2009.

[37] P. W. Holland and S. Leinhardt. A method for detecting structure in sociometric data. *American Journal of Sociology*, 76:492–513, 1970.

[38] M. O. Jackson, T. Rodriguez-Barraquer, and X. Tan. Social capital and social quilts: Network patterns of favor exchange. *American Economic Review*, 102(5):1857–1897, 2012.

[39] S. Jain and C. Seshadhri. A fast and provable method for estimating clique counts using turán's theorem. In *Conference on the World Wide Web (WWW)*, pages 441–449, 2017.

[40] T. Kloks, D. Kratsch, and H. Müller. Finding and counting small induced subgraphs efficiently. *Information Processing Letters*, 74(3-4):115–121, 2000.

[41] T. Kopelowitz, S. Pettie, and E. Porat. Higher lower bounds from the 3sum conjecture. In *Symposium on Discrete Algorithms (SODA)*, pages 1272–1287, 2016.

[42] S. Marko and D. Ron. Approximating the distance to properties in bounded-degree and general sparse graphs. *ACM Transactions on Algorithms*, 5(2):22, 2009.

[43] G. Marsaglia, W. W. Tsang, J. Wang, et al. Fast generation of discrete random variables. *Journal of Statistical Software*, 11(3):1–11, 2004.

[44] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon. Network motifs: simple building blocks of complex networks. *Science*, 298(5594):824–827, 2002.

[45] M. Mitzenmacher, J. Pachocki, R. Peng, C. E. Tsourakakis, and S. C. Xu. Scalable large near-clique detection in large-scale networks via sampling. In *SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 815–824, 2015.

[46] C. St. JA. Nash-Williams. Edge-disjoint spanning trees of finite graphs. *Journal of the London Mathematical Society*, 1(1):445–450, 1961.

[47] C. St. JA. Nash-Williams. Decomposition of finite graphs into forests. *Journal of the London Mathematical Society*, 1(1):12–12, 1964.

[48] J. Nešetřil and P. Ossana de Mendez. *Sparsity: Graphs, Structures, and Algorithms*. Springer, 2012.

[49] J. Nešetřil and S. Poljak. On the complexity of the subgraph problem. *Commentationes Mathematicae*

Universitatis Carolinae, 26(2):415–419, 1985.

[50] H. N. Nguyen and K. Onak. Constant-time approximation algorithms via local improvements. In *Foundations of Computer Science (FOCS)*, pages 327–336, 2008.

[51] K. Onak, D. Ron, M. Rosen, and R. Rubinfeld. A near-optimal sublinear-time algorithm for approximating the minimum vertex cover size. In *Symposium on Discrete Algorithms (SODA)*, pages 1123–1131, 2012.

[52] M. Parnas and D. Ron. Approximating the minimum vertex cover in sublinear time and a connection to distributed algorithms. *Theoretical Computer Science*, 381(1-3):183–196, 2007.

[53] A. Portes. Social capital: Its origins and applications in modern sociology. In Eric L. Lesser, editor, *Knowledge and Social Capital*, pages 43 – 67. Butterworth-Heinemann, Boston, 2000.

[54] C. Seshadhri, T. G. Kolda, and A. Pinar. Community structure and scale-free collections of Erdös-Rényi graphs. *Physical Review E*, 85(5):056109, May 2012.

[55] K. Shin, T. Eliassi-Rad, and C. Faloutsos. Patterns and anomalies in k-cores of real-world graphs with applications. *Knowledge and Information Systems*, 54(3):677–710, 2018.

[56] S. Suri and S. Vassilvitskii. Counting triangles and the curse of the last reducer. In *Proceedings of the International Conference on World Wide Web (WWW)*, pages 607–614, 2011.

[57] C. E. Tsourakakis. The k-clique densest subgraph problem. In *Proceedings of the International Conference on World Wide Web (WWW)*, pages 1122–1132, 2015.

[58] V. Vassilevska. Efficient algorithms for clique problems. *Information Processing Letters*, 109(4):254–257, 2009.

[59] A. J. Walker. New fast method for generating discrete random numbers with arbitrary frequency distributions. *Electronics Letters*, 10(8):127–128, 1974.

[60] A. J. Walker. An efficient method for generating discrete random variables with general distributions. *ACM Transactions on Mathematical Software*, 3(3):253–256, 1977.

[61] Y. Yoshida, M. Yamamoto, and H. Ito. An improved constant-time approximation algorithm for maximum. In *Proceedings of the Symposium on Theory of Computing (STOC)*, pages 225–234, 2009.